PCT/SE 00 / 0 1 7 4 4

PRV

PATENT- OCH REGISTRERINGSVERKET
Patentavdelningen

SE00/1744

07 -09- 2000

REC'D 13 OCT 2000
WIPO PCT

EU

Intyg
Certificate

10707070 1

Härmed intygas att bifogade kopior överensstämmer med de handlingar som ursprungligen ingivits till Patent- och registreringsverket i nedannämnda ansökan.

This is to certify that the annexed is a true copy of the documents as originally filed with the Patent- and Registration Office in connection with the following patent application.

(71) Sökande *Pacesetter AB, Järfälla SE*
Applicant (s)

(21) Patentansökningsnummer *9903191-6*
Patent application number

(86) Ingivningsdatum *1999-09-08*
Date of filing

Stockholm, 2000-04-10

CERTIFIED COPY OF PRIORITY DOCUMENT

För Patent- och registreringsverket
For the Patent- and Registration Office

A. Södervall
Anita Södervall

Avgift
Fee 170:-

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Compression and decompression coding scheme and apparatus

Field of invention

The invention relates generally to a coding scheme and apparatus for the compression and decompression of data. It is particularly directed to the compression of signals that exhibit so-called memory, where a portion of a signal depends on the value of a preceding portion. The invention has particular application to medical systems, such as implantable pacemaker devices, which have limited memory but require the storage of large quantities of data.

Background art

Medical systems for the monitoring of physiological functions are becoming more complex as the need for diagnostic applications increases. In particular there is a need for intracardial detection systems and pacemaker systems capable of storing ever increasing numbers of signals, such as electrocardiagram signals (ECG and IEGM), pressure signals and bioimpedence signals or the like, of ever increasing length. However, the available memory space for data storage is often restricted, particularly in implanted pacemaker systems. Perhaps more importantly in implanted systems, the amount of data that may be collected is also restricted by the transmission capacity of a telemetry link between an implanted device and its programmer or other external control device. For example, a defibrillator today typically requires a transmission time of up to 40 minutes for downloading to its controller all the data that can be collected. If the required quantities of data are to be made available for processing, this data must be compressed.

Data compression can generally be divided into two forms. A first form is

based on viewing a signal as a mathematical function and observing and utilising characteristics of this function to compress data. The second form utilises coding theory and is based on the statistics of multiple discrete signal levels, or symbols, occurring in a signal.

A conventional algorithm working according to this latter principle is the Tunstall code. This code maps variable-length symbol segments of an input signal into fixed-length codewords. Since the codeword length is fixed, the number of codewords n is known in advance. The object is to assign codewords to symbol segments that occur with approximately equal probability. The procedure begins with a set of symbol segments consisting of each of the individual symbols occurring in the input signal, say m symbols in all. The most probable symbol is then removed from the set and replaced by m new segments, each of which is the removed symbol suffixed by one of the m input symbols. This procedure continues until the number of symbol segments, m, in the set is equal to the number of codewords, n. The codewords are then assigned to the symbol segments.

An example of Tunstall encoding is illustrated in Figs. 1 and 2. Fig. 1 shows a probability tree comprising nodes, and branches emanating from each node. In the illustrated example it is assumed that 8 codewords are available for compressing data, 1 to 8. The tree in Fig. 1 assumes that a source signal comprises two signals, ‘0’ and ‘1’. Each symbol is represented by a first branch. The probability a symbol occurring is given at the node terminating the associated branch. Hence a ‘0’ occurs with a probability of 0.6 and a ‘1’ occurs with a probability of 0.4. The branch with the highest probability is expanded further by adding a second series of branches for each possible symbol. The ‘0’ branch is thus bifurcated into a second ‘0’ branch with a total probability of 0.36 and a ‘1’ branch with probability 0.24. The first branch for symbol ‘1’ now has the highest probability and is expanded in turn, resulting

in two further branches with probability of 0.24 and 0.16. This process continues until the number of end branches equals the number of codewords, in the present example 8, with the probability of each branch occurring being relatively close. Each end branch represents a sequence of symbols. These sequences are assigned a codeword as shown in Fig. 2.

A problem associated with Tunstall encoding is that signals containing a large number of different symbols, for instance a large number of discrete signal levels, require a very large number of codewords. For example, in a pacemaker system, a typical electrogram or IEGM signal is represented by 8 bits sampled at 512 Hz. The number of symbols in this signal is thus 256. A Tunstall codebook for this signal would have to contain at least 256 codewords just to cover the different individual symbols. In order to obtain compression of the signal, the branches must be expanded further which adds a further 256 codewords per branch. The Tunstall code is a general-purpose data compression algorithm and is not adapted to special classes of data. In particular it is not effective when applied to signals which exhibit memory. Typically, many of the signals monitored by medical systems, and cardiac pacers in particular, exhibit some memory, one example being the IEGM signal. The use of Tunstall encoding for processing sampled data in medical systems, and specifically implanted systems, is thus limited.

In the light of this prior art, it is an object of the present invention to provide a coding method and apparatus that allows a signal exhibiting memory to be compressed and decompressed efficiently and without distortion.

SUMMARY OF INVENTION

Viewed from one aspect, the present invention proposes a data coding method for converting a signal containing a plurality of symbols into codewords, including the steps of : establishing a set of codewords, observing a data signal

and determining the most frequently occurring symbols and/or sequences of symbols containing at least two symbols, allocating one codeword to each of the most frequently occurring of said symbols and/or symbol sequences.

According to a further aspect, the invention provides a data compression method for compressing a data signal containing a plurality of symbols, including: converting the most frequently occurring symbols and/or symbol sequences into codewords, supplementing the remaining symbols with at least one codeword indicative of no compression.

The invention further proposes an arrangement for compressing and decompressing a data signal containing a plurality of symbols, including: means for storing codewords corresponding to symbols and/or symbol sequences, and means for determining if a symbol in said data signal corresponds to at least one codeword in the storage means and, when a symbol corresponds to only one codeword, for transmitting said codeword, wherein the determining means are further adapted to transmit a symbol if it corresponds to no codeword in the storage means.

According to a fourth aspect, the invention proposes a computer program product for converting a signal containing a plurality of symbols into a compressed signal, including computer readable program code means for establishing a set of codewords, determining the most frequently occurring symbols and/or sequences of symbols containing at least two symbols in a data signal and allocating one codeword to each of the most frequently occurring of said symbols and/or symbol sequences.

By providing codewords to only the most frequently occurring symbols and symbols sequences, the number of codewords required is greatly reduced. At the same time however the efficiency of the compression is increased, as these

codewords are allocated to the varying length symbol sequences that appear with the highest frequency in the signal. This compression technique furthermore fully exploits any memory in a signal. A further difference over prior art coding schemes and the Tunstall code in particular is that codewords may be allocated to every node and end branch in a coding tree and not just the end branches. This is also particularly useful when compressing signals that exhibit memory such as signals monitoring physiological quantities such as the heartbeat, respiration rate and the like.

BRIEF DESCRIPTION OF THE DRAWINGS

Further objects and advantages of the present invention will become apparent from the following description of the preferred embodiments that are given by way of example with reference to the accompanying drawings, in which:

Fig. 1	depicts a coding tree illustrating the Tunstall encoding algorithm;
Fig. 2	depicts a coding table corresponding to the coding tree of Fig. 1
Fig. 3	depicts a schematic block diagram of an encoder/decoder (compressor/decompressor) according to the present invention.
Figs. 4a To 4e	depicts a series of histograms illustrating the generation of a codebook according to the present invention;
Fig. 5	depicts a coding tree corresponding to the histograms of Fig. 3;
Fig. 6	depicts a codebook corresponding to the tree of Fig. 4;
Fig. 7	is a schematic illustration of a pre-processing function in

accordance with the present invention.

Fig. 8	is a schematic diagram showing an arrangement for compressing and decompressing data according to the present invention;
Fig. 9	is a schematic illustration of a codebook memory illustrating the mapping between symbols and memory locations;
Fig. 10	is a flow chart illustrating a method for compressing data utilising the arrangement of Fig. 8; and
Figs 11a to 11e	show a sequence of graphs illustrating the compression and subsequent decompression of an IEGM signal.

DETAILED DESCRIPTION OF THE DRAWINGS

The compressing and decompression scheme according to the present invention is based on the representation of a data signal containing a plurality of symbols into coded form utilising codewords of fixed length. The symbols contained in an input signal may be digital representations of characters, such the ASCII format. Typically, however, the symbols will be binary representations of discrete signal levels in a sampled analogue signal, such as an ECG, IEGM or other signals for monitoring physiological activity.

Fig. 3 schematically illustrates the function of an encoder/decoder for compressing and decompressing a data signal. The data signal contains a plurality of symbols which are read by a symbol reader 1 and relayed to an encoder/decoder 2. The encoder/decoder 2 has access to a storage medium 3 containing a codebook of the fixed-length codewords. The input symbols or symbol sequences are mapped to codewords in the codebook 3 and replaced by the corresponding codeword. The reduction of symbols and symbol

sequences of variable length into codewords results in the compression of the data. Decompression is accomplished by inverting the operation. Codewords are passed through the encoder/decoder 2, which with the aid of the mapping to the codebook 3 reconstitutes the original input sequence.

The generation of a codebook according to the present invention commences with determining the number of codewords to be used. This is selected as a function of the total number of symbols contained in an input signal and the degree of compression required.

The codebook is generated by observing the input signal during a test phase and determining the probabilities of the symbols and sequences of symbols occurring. This is accomplished by observing the input signal during a set time period, for example 20 s, and noting the symbol that occurs with the highest frequency during this period. The signal is then observed again for the same time period and the symbol that follows the noted symbol most frequently is noted. This process continues until the number of most frequently occurring symbols and/or symbol sequences noted is equal to the number of codewords in the codebook. It will be understood, that only those symbols or symbol sequences that occur most frequently will be coded. This allows the number of codewords to be kept to a reasonable number.

A simplified example of this process is illustrated in Figs. 4, 5 and 6 with reference to the generation of a codebook containing 8 symbols. The input signal consists of 5 symbols 1,2,3,4,5.

Turning now to Fig. 4a, a first histogram is produced after observing the input signal for a set period. It is evident that the most frequently occurring symbol is 3 with an occurrence of 50. Fig. 4b shows a histogram produced after the same period of time and gives the frequency of the different symbols that

follow 3. Thus the sequence 32 has occurred ten times, the sequence 33 fifteen times, and the sequence 34 twelve times. Fig. 4c shows a histogram of the symbols following 2 and Fig. 4d shows a histogram of the symbols following the symbol sequence 3, 3. Finally Fig. 4e shows the histogram of the symbols following the symbol 4. The codebook is established by determining which of the 8 symbols and/or symbol sequences occurred most frequently. This is illustrated schematically in a code tree in Fig. 5. In this code tree, a codeword has been assigned to every node and end branch. Hence the symbol 3 has been allocated the codeword '2' but the symbol sequence 33 has also been allocated a codeword, '5' and the symbol sequence 332 has been allocated the codeword '8'.. This is summarised in tabular form in Fig. 6.

It is apparent from the above example that codewords are not assigned to every symbol of an input signal. Thus when these unassigned symbols are read by an encoder, the symbol is transmitted uncompressed. To distinguish the uncompressed data from the compressed data, it is preferably to utilise some form of distinguishing symbol or codeword. Thus at least one codeword will be reserved for transmitting uncoded input symbols.

Depending on the signal to be compressed, it may be desirable to pre-process the signal in order to increase the frequency of a very few symbols and sequences. This considerably increases the efficiency of the compression. For example, pre-processing can be very effective when compressing sampled IEGM, ECG or other physiological signals that monitor some form of periodic activity.

Fig. 7 schematically illustrates the preferred function of such a pre-processor. In Fig. 7 an input data stream denoted by 10 comprises the symbols IEGMn-1, IEGMn, IEGMn+1 and IEGMn+2. The function generates the difference symbol value between a symbol and a preceding symbol. The output data

stream 12 thus comprises the symbols (IEGMn - IEGMn-1) and (IEGMn-1 – IEGMn-2), etc.

In a preferred embodiment of the present invention, the compression coding scheme according to the invention is used to compress an IEGM signal comprising 255 symbols ranging from 0 to 254. After pre-processing in accordance with the function of Fig. 7, the signal contains 509 possible symbols, ranging from –254 to +254. However, while the number of possible symbols has almost doubled, the form of the original IEGM signal is such that the processed signal contains mainly symbols close to 0 such as 1, -1, 2, -2 etc. Thus the concentration of a very few symbols and symbol sequences is increased by the difference function.

A codebook generated for a pre-processed IEGM signal of the type described above can typically be efficiently compressed utilising a codebook containing 254 codewords, for example 254 8-bit words ranging from 0 to 253. The 254 most frequently occurring symbols and/or symbol sequences are then converted into codewords. The 8-bit words with values 254 and 255 are then reserved. These are utilised to signal whether data is compressed or uncompressed.. Preferably, one reserved codeword, for example 255, is sent to indicate that uncompressed data follows. The other codeword 254 can then be utilised to signal that compressed data is following. To avoid having to generate different symbols for negative and positive symbols, an uncompressed negative symbol is preferably indicated by preceding it with both symbols sent contiguously, for example, 255 followed by 254 followed by the uncompressed equivalent positive symbol.

The codebook is preferably generated for a class of signals and retained for different signals of the same class so that it does not need to be re-established prior to compressing data. For example, a cardiac pacer containing data

compression and decompression circuitry for processing some physiological measurement such as an IEGM signal, ECG signal or bioimpedence signal could be subjected to a training phase for each patient to establish a codebook that is specific to each patient. The training phase could also be performed using a test sequence that is representative of the class of signal, so that one codebook is used for several patients. A further option would be to utilise an adaptive procedure, whereby the statistics of the signal are observed and a codebook newly generated prior to each compression. This codebook would then be optimised for each signal generated in a pacemaker. However, it will be understood that the codebook must be retained long enough to enable the compressed data to be decompressed.

In the coding procedures described above, a codeword is allocated to every node and end branch of a probability tree. It will, however, be understood that codewords could be assigned only to the end branches of the tree. The efficiency of such a code would depend on the characteristics of the starting signal.

An arrangement for encoding and decoding a signal is illustrated in Fig. 8. The arrangement includes an input stage 20 for reading a symbol. A processor 21, that preferably takes the form of a single chip microprocessor with associated memory, is coupled to the input stage. A codebook memory 22 having 256 8-bit memory locations with addresses from 0 to 255 is connected to the processor 21. An output stage 23 for the codewords is coupled to the memory 22 and processor 21. This output stage 23 is finally coupled to a storage memory 24 for storing the compressed data and to a telemetry transmission unit 26, which forwards the data to a remote external programmer or controller. The input stage 20 is also coupled to the storage memory 24 and telemetry transmission unit 26 for transmitting uncompressed data. The encoding and decoding arrangement is preferably preceded by a pre-

processing stage as shown in Fig. 7.

The function of this arrangement is basically as follows. A first symbol is read by the input stage 20. The processor 21 checks whether this symbol corresponds to more than one symbol sequence in the codeword memory 22 and if so, the next symbol is read. This process is repeated until the sequence of symbols read corresponds to only one codeword. This codeword is then emitted in place of the symbol sequence and is either stored in the memory 24 or sent directly to an external device via the telemetry transmission unit 26. If a symbol read corresponds to no coded sequences it is transmitted unchanged to the storage memory 24 or transmission unit 26 but preceded by the codeword 254 indicating uncompressed data.

The addresses 0 to 255 of the codebook memory 22 are the codewords. The processor furthermore performs a mapping between the incoming symbols that form at least part of a coded sequence and each memory cell 221. This is illustrated in more detail in Fig. 9. The first symbol in a sequence will cause the processor to access the first memory cell, in ascending order of address, to which the symbol is mapped. These first memory cells 221 correspond to the first branches in a probability tree. Thus if the first symbol in a sequence is '1', the processor 21 will access the fourth memory location (address 3), since this is the first location 221 to which a '1' is mapped. Each memory cell 221 further contains information indicating whether the mapped symbol corresponds to more than one coded symbol sequence. This information is shown on the right hand side of each memory cell 221. The information is in the form of a code represented by the numbers 0 to 7, which indicate both the number of possible further branches and the following symbols corresponding to the further branches. A conversion table 25 shows the significance of each number. Hence it can be seen that '0' indicates an end branch, '2' represents two possible further branches with the following symbols '1' and '2'. Any

symbol mapped to a memory cell 221 containing a '0' will result in the processor transmitting the address of the memory cell as a codeword. Any other number contained in the memory cell 221 indicates that at least one further branch is possible. The processor 21 then fetches the next symbol from the input stage 20 and determines whether this symbol forms part of the possible codewords. If it does, the processor calculates the address of the next memory cell 221 and the process is repeated.

The next address is calculated by summing the number of possible branches contained in all addresses starting from the first address up to the present address, and then adding the position of the fetched symbol in the list of possible symbols given in the conversion table 25. The sum of possible branches is equal to the sum of the first mapped memory locations representing the initial branches of the probability tree and the possible branches stored in all previous memory locations. Thus an input sequence consisting of 1, 2 would result in a first mapping being made to memory location 3. This contains the code 5, which indicates that four further branches are possible. The next symbol is fetched. It is verified that '2' corresponds to one of the possible branches. The symbol '2' has the fourth position in the list of branches (–1, 0, 1, 2) given in the conversion table 25. Thus the sequence 1, 2 is a valid coded sequence. The next address is equal to the sum of the initial basic branches (i.e. the addresses to which a first symbol can be mapped), which in the present example is 5 (addresses 0 to 4). To this is added the sum of the branches contained in memory locations up to address 3, and the position of the subsequent symbol in the list indicated in memory location 3. This gives a total of $5 + (3 + 1 + 2) + 4 = 15$. The next address is thus the 15^{th} location or address 14, since the addresses start with 0. This is verified by the mapping of the symbol 2 indicated in the left-hand column of the cell with address 14.

The conversion table 25 may alternatively be more complex in structure and provide absolute memory locations corresponding to each possible branch. In this way the next address would not need to be calculated, but more storage capacity would be needed.

The conversion table is preferably stored in the processor 21. However, it may be possible to store some of the information about the branches in the codeword memory, depending on how much capacity is available for each address.

The arrangement illustrated in Fig. 8 is a schematic representation of possible encoding and decoding hardware. It will however, be understood that the functions of the various elements shown in Fig. 8 may be performed entirely in a digital processor system operating under the control of a program. The codebook memory could then be implemented virtually from part of the memory space incorporated in the processor system.

Fig. 10 is a flow chart illustrating the procedure for compressing data using the arrangement shown in Figs. 8 and 9. The procedure starts at step 30 with the reading of a symbol. In step 31 it is determined whether the symbol is inside the designated range, i.e. whether the symbol forms part of a coded sequence and can be mapped to a memory location. If this is the case, the process moves to step 32 and the new memory address is calculated. In the following step 33 a marker, 'codeword_started', indicating that the coding of a sequence has started, is set. In step 34 it is verified whether the data last sent was uncompressed. If this is true, in step 35 the symbol indicating compression is sent. If the last data sent was in compressed mode, the process moves directly from step 34 to step 36, where the contents of the memory location are read and it is determined whether an end branch has been reached. If the end branch has not been reached the procedure returns to step 30 and the next

symbol is fetched. If the end branch is reached, the procedure passes to step 38, where the memory address is sent as the codeword. In step 39, the memory address is reset to 0 and in step 40, the marker 'codeword_started' is reset to false, because the coding of a symbol or symbol sequence is terminated. The process then returns to step 30, and the next symbol fetched. If a symbol is discovered to be out of range in step 31, indicating that the symbol does not form part of a coded sequence, the procedure goes to step 41 where the status of the marker 'codeword_started' is verified. If this is true, this means that a codeword has been started, but the subsequent symbol does not form part of the coded sequence. In step 42, therefore, the memory address is sent as the codeword. The memory address is then reset to 0 in step 43, the marker 'codeword_started' reset to false in step 44 and the procedure returns to step 31, where the fetched symbol is checked against the starting symbols of coded sequences to verify if it forms part of a coded sequence. If in step 41, it is determined that no codeword has been started, i.e. the status of the 'codeword_started' marker is false, this means that the fetched symbol is not contained in any coded sequence. In step 45, the transmission mode is checked. If the last data sent was compressed, the symbol indicating uncompressed data is sent in step 46 followed by the read symbol in step 47. If the last transmission was not in compression mode, the symbol is sent in step 47. The procedure then returns to the start at step 30.

Decompression is the exact reverse of the compression procedure described above. Each codeword is converted into the corresponding symbol or symbol sequence. The symbols are then summed to retrieve the original uncompressed data.

Figs. 11a to 11c illustrate an example using the coding algorithm described above. The algorithm was first trained, i.e. the codebook generated, using an IEGM signal containing 10000 samples sampled at 512 Hz with 8-bit

resolution. The training signal is illustrated in Fig. 11a. The algorithm was then tested on an IEGM signal containing 10000 samples sampled at 512 Hz with 8-bit resolution. This uncompressed test signal is depicted in Fig. 11b. Fig. 11c shows the signal after compression. This signal contains 2149 samples which gives a compression ratio of 4.6. Fig. 11d shows the signal of Fig. 11c after decompression. Finally, Fig. 11e shows the difference signal between the original signal depicted in Fig. 11b and the decompressed signal of Fig. 11d. The signal is entirely free of distortion.

In the coding procedures described above, a codeword is allocated to every node and end branch of a probability tree. It will, however, be understood that codewords could be assigned only to the end branches of the tree.

P000696 SE/HG

Claims:

1. A data coding method for converting an input signal containing a plurality of symbols into a compressed signal, including establishing a set of codewords, the method being characterised by the steps of: observing a data signal and determining the most frequently occurring symbols and/or sequences of symbols containing at least two symbols, allocating one codeword to each of the most frequently occurring of said symbols and/or symbol sequences.

2. A method as claimed in claim 2, characterised in that each determining step is performed by observing the data signal during a predetermined time period.

3. A method as claimed in claim 1 or 2, characterised by reserving at least one codeword in the set to serve as indicator for uncoded symbols.

4. A method as claimed in any one of claims 1 to 3, characterised by allocating codewords to symbols and symbol sequences that are incorporated in other symbol sequences having an allocated codeword.

5. A method as claimed in any one of claims 1 to 4, characterised by supplementing uncoded negative symbols with at least one codeword indicative of a negative value.

6. A data compression method for compressing a data signal containing a plurality of symbols, characterised by including: converting the most frequently occurring symbols and/or symbol sequences into codewords,

supplementing the remaining symbols with at least one codeword indicative of no compression.

7. A method as claimed in claim 6, wherein the number and length of the codewords are predetermined.

8. A method as claimed in claim 6 or 7, including pre-processing an input signal containing a plurality of symbols to generate said data signal by generating a symbol representing the difference between contiguous symbols.

9. A method as claimed in any one of claims 6 to 8 further characterised by
reading a symbol,
determining if said symbol corresponds to at least one codeword, and
outputting a codeword if said symbol corresponds to only said one codeword.

10. A method as claimed in claim 9, characterised in that when a symbol corresponds to more than one codeword, reading a subsequent symbol, determining if said symbol corresponds to at least one codeword, and outputting a codeword if said symbol corresponds to only said one codeword.

11. A method as claimed in claim 9 or 10, characterised by
when a symbol corresponds to no codeword, outputting said symbol.

12. An arrangement for compressing and decompressing a data signal containing a plurality of symbols, characterised by including:
means (22) for storing codewords corresponding to symbols and/or

symbol sequences, and
means (20, 21, 23, 25) for determining if a symbol in said data signal corresponds to at least one codeword in said storage means and, when a symbol corresponds to only one codeword, for transmitting said codeword, wherein the determining means are further adapted to transmit a symbol if it corresponds to no codeword in said storage means.

13. An arrangement as claimed in claim 12, characterised in that said storage means (22) include a plurality of storage locations (221) designating codewords, wherein each storage location (221) contains an indication of the number of possible coded symbol sequences, and is mapped to a symbol of said data signal.

14. An arrangement as claimed in claim 12 or 13, characterised in that means generating a difference symbol between contiguous symbols in said data signal are connected upstream of said determining means (20, 21, 23, 25).

15. An arrangement as claimed in any one of claims 12 to 14, characterised in that the addresses of the storage locations (221) are codewords.

16. A computer program product for converting a signal containing a plurality of symbols into a compressed signal, characterised by including computer readable program code means for establishing a set of codewords, determining the most frequently occurring symbols and/or sequences of symbols containing at least two symbols in a data signal and,
allocating one codeword to each of the most frequently occurring of said symbols and/or symbol sequences.

17. A program product as claimed in claim 16, further characterised by computer readable program code means for compressing said data signal by converting the most frequently occurring symbols and/or symbol sequences into said codewords by:

reading a symbol,

determining if said symbol corresponds to at least one codeword, and outputting a codeword if said symbol corresponds to only said one codeword, and by outputting a symbol when said symbol corresponds to no codeword,

P000696 SE/HG

Abstract

The invention relates to a compression and decompression coding scheme and arrangement. The scheme converts a data signal containing a plurality of symbols into a series of codewords. A set of codewords is established and the data signal is subsequently observed to determining the most frequently occurring symbols and/or sequences of symbols containing at least two symbols. A codeword is then allocated to each of the most frequently occurring of said symbols and/or symbol sequences. At least one codeword is reserved for indicating uncompressed data. When compressing a signal, the incoming symbols are first checked to see if they correspond to at least one codeword. If a symbol corresponds to more than one codeword, further symbols are read until a symbol is found which corresponds to one codeword only. The codeword is then transmitted. Any symbols that correspond to no codewords are supplemented with a codeword indicative of no compression and then transmitted.

Fig. 10

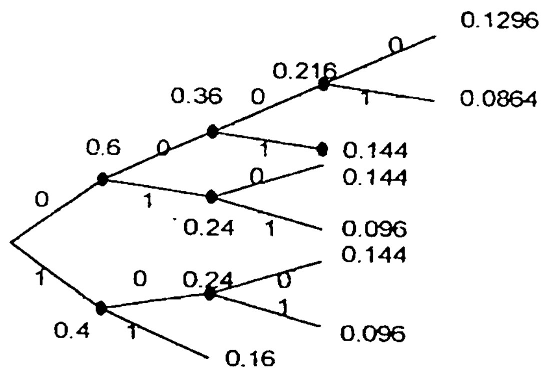

Fig. 1

Symbol sequences	Codeword	Codeword binary form
0000	1	001
0001	2	010
001	3	011
010	4	100
011	5	101
100	6	110
101	7	111
11	8	000

Fig. 2

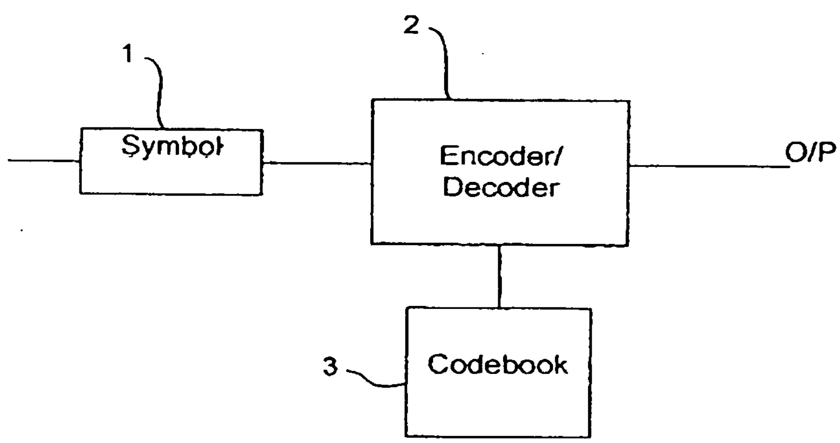

Fig. 3

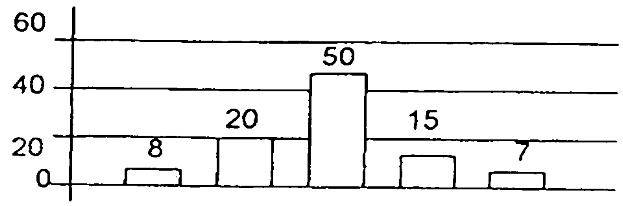

Fig. 4a

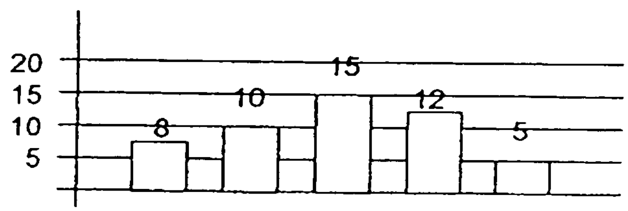

Fig. 4b

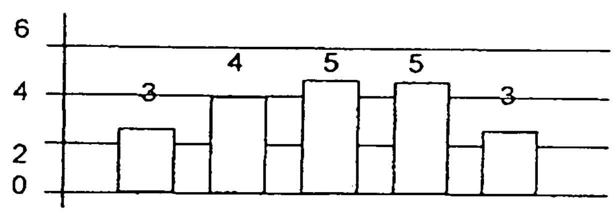

Fig. 4c

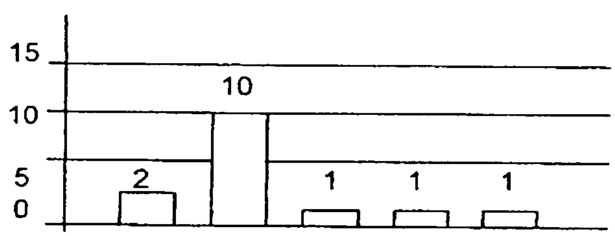

Fig. 4d

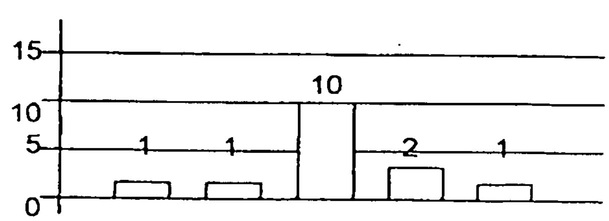

Fig. 4e

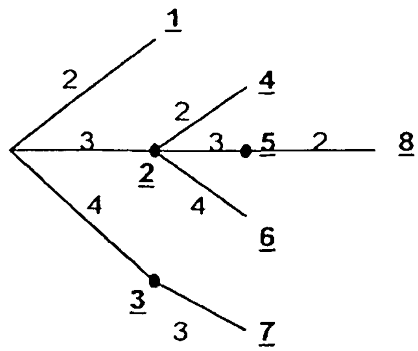

Fig. 5

Symbol sequence	Codeword
2	1
3	2
4	3
32	4
33	5
34	6
43	7
332	8

Fig. 6

25

Code No.	Possible branches	No. of branches
0	End	0
1	0	1
2	-1, 0	2
3	0, 1	2
4	-1,0,1	3
5	-1,0,1,2	4
6	-2,-1,0,1	4
7	-2,-1,0,1,2	5

Fig. 9

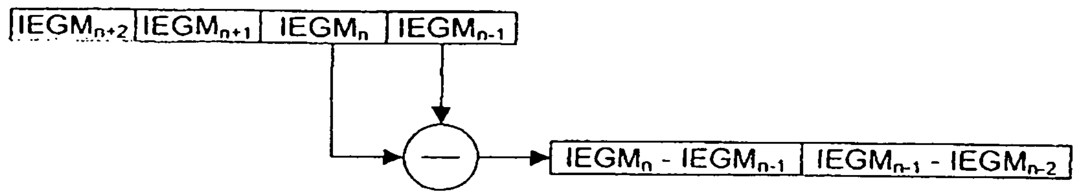
$IEGM_{n+2}$
$IEGM_{n+1}$
$IEGM_n$
$IEGM_{n-1}$
—
$IEGM_n - IEGM_{n-1}$
$IEGM_{n-1} - IEGM_{n-2}$

Fig. 7

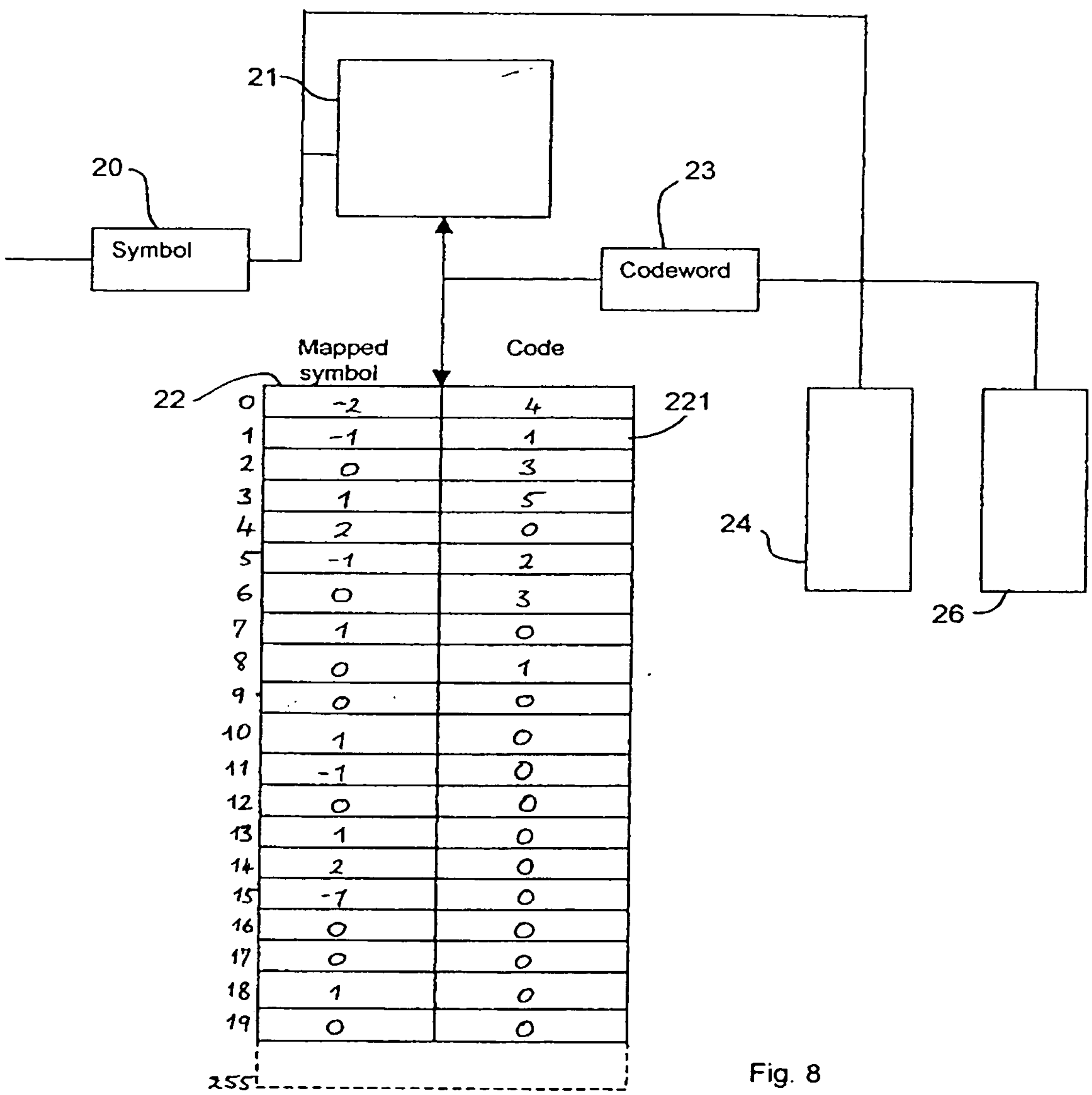
21
20
Symbol
23
Codeword
Mapped symbol
Code
22
221
24
26
0 -2 4
1 -1 1
2 0 3
3 1 5
4 2 0
5 -1 2
6 0 3
7 1 0
8 0 1
9 0 0
10 1 0
11 -1 0
12 0 0
13 1 0
14 2 0
15 -1 0
16 0 0
17 0 0
18 1 0
19 0 0
255

Fig. 8

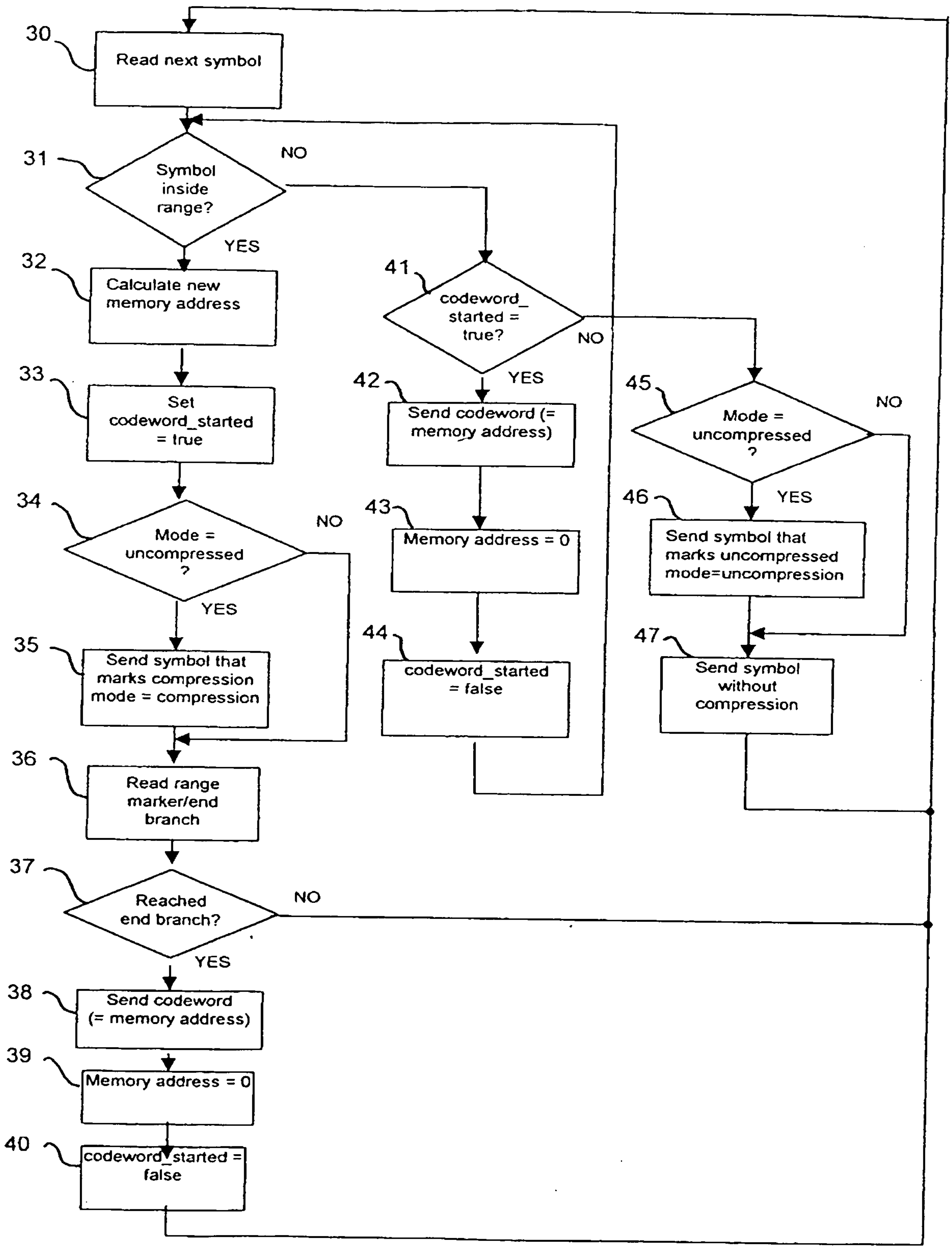
30
Read next symbol
31
Symbol inside range?
NO
YES
32
Calculate new memory address
33
Set codeword_started = true
34
Mode = uncompressed ?
NO
YES
35
Send symbol that marks compression mode = compression
36
Read range marker/end branch
37
Reached end branch?
NO
YES
38
Send codeword (= memory address)
39
Memory address = 0
40
codeword_started = false
41
codeword_ started = true?
NO
YES
42
Send codeword (= memory address)
43
Memory address = 0
44
codeword_started = false
45
Mode = uncompressed ?
NO
YES
46
Send symbol that marks uncompressed mode=uncompression
47
Send symbol without compression

Fig. 10

Fig. 11a

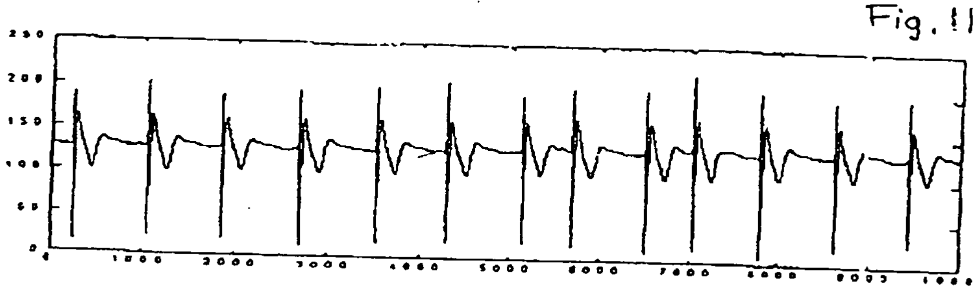

Fig. 11b

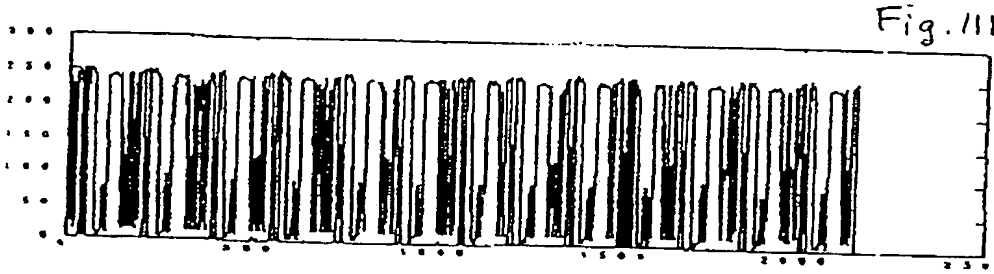

Fig. 11c

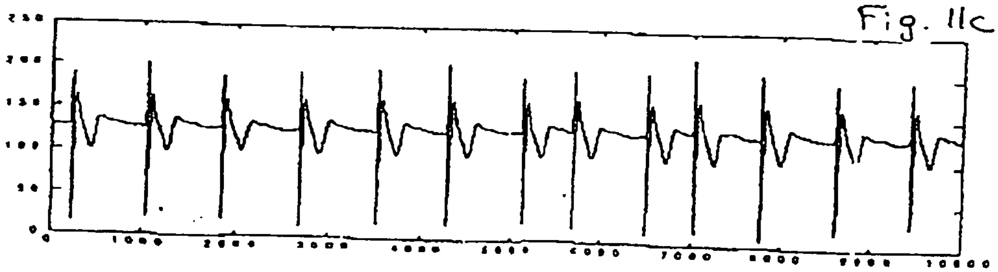

Fig. 11d

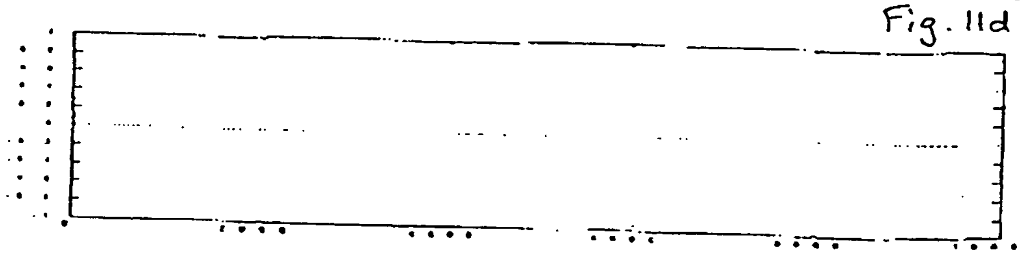

Fig. 11e

10/070701

JC19 Rec'd PCT/PTO 07 MAR 2002

CERTIFICATION OF MAILING

"Express Mail" Mailing Label Number **# EJ 552525055US**

Date of Deposit: March 7, 2002

I hereby certify that this correspondence is being deposited with the United States Postal "Express Mail Post Office to Addressee" service under 37 C.F.R. §1.10 on the date indicated above and is addressed to: Assistant Commissioner for Patents and Trademarks, Washington, D.C. 20231. **BOX PCT**

Johan Lidman

Patent Application entitled, **"COMPRESSION AND DECOMPRESSION CODING SCHEME AND APPARATUS"**

Our Case No. P02,0086 (27009-0125)

Signature of person mailing application and fee